КУРСОВ ПРОЕКТ

Онтология на криминална случка

Част 2

Факултет по Математика и Информатика

Студент: Борислав Стоянов Марков

Факултетен номер: 0MI3400048

Учебен план: Изкуствен Интелект (редовно, магистър)

Курс: **Курс 1**; Група: **Група 1**

Активен период: 2021/2022 летен, магистри

Дисциплина: Бази от знания

Дата: 12.06.2021г.

1. Съдържание

1. Съдържание 2

2. Увод 2

3. Сапунена криминална случка 3

4. Онтология на затворен свят 4

4.1 Елементи на онтологията 4

4.2. Изводи със SPARQL[4] към базата 8

4.3. Анализ на резултатите 16

5. Реализация на проекта 16

6. Идеи за бъдещо развитие и подобрения 17

7. Източници и използвана литература 17

Приложения 18

1. Сорс код (Source code) 18

2. Увод

Идеята за проекта е взета от сапунената опера от книгата „Knowledge Representation and Reasoning“ (Brachman и Levesque)[1] и е допълнение на онтологията направена от първи семестър. Проектът моделира събитие от измислен свят. В града се е случила криминална случка. Има изтичане на информация от една софтуерна фирма и градският детектив разпитва заподозрените. Предстои да опишем това с онтология. Ще проследим кой кого познава и така ще помогнем на детектива да стигне до заподозрените. Описателният език, който ще използваме е OWL DL[3], използваната база за графи е GraphDB[2]. Ще се използва машината за изводи (Reasoning) предлагана от OWL и GraphDB. В този проект няма да използваме наши правила, както е в предишният проект, а изводите ще се правят на база на SPARQL заявки. Всички данни са въведени ръчно (поради неголемият обем) и са в TURTLE формат. Така ще бъдат импортирани в базата

GraphDB, а в последствие ще направим няколко извода със SPARQL от уеб интерфейса предлаган от базата.

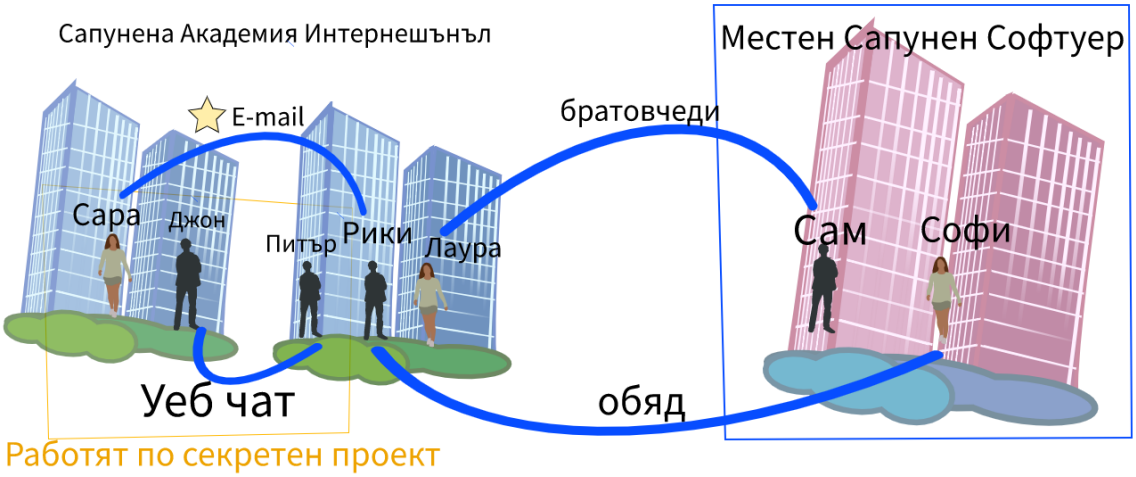
Целите на проекта са да се покаже как може да се моделират знанията от реалния свят и да се правят изводи, когато това не е очевидно. Областта може да е криминалистика, но не се ограничаваме само до нея.

3. Сапунена криминална случка

Джон(John) работи във софтуерната фирма „Сапунена Академия Интернешънъл“ (Soap Academy International). Проектът на Джон е конфиденциален и се казва „Робо-Сапунена Стратегия“ (Robo Soap Strategy). Джон има колега, с когото добре се познават, който се казва Питър (Peter) и работят по един проект. Пишат си постоянно във фирмения чат. Джон и Питър имат колега Рики (Riki), който се познава с Питър, но не се познават с Джон лично. Джон има също и колежка Сара(Sara), която работи по същият проект като него. Рики не работи по проекта „Робо-Сапунена Стратегия“. Лаура тайно харесва Джон, но е в друг офис, не работят заедно и се виждат само по фирмени събирания или на онлайн срещи. Джон не харесва Лаура и не откликва на чувствата й. От друга страна Рики харесва Лаура, която не му обръща внимание. Рики е готов на всичко да я заинтереува. От друга фирма „Местен Сапунен Софтуер“ (Domestic Soap Software), конкурент на Сапунена Академия Интернешънъл, са получили информация за проекта Робо-Сапунена Стратегия и информацията е обявена публично. Местният детектив разпитва участниците за случките през последната седмица и установява, че Джон не работи в една сграда с Питър. Питър работи в една сграда с Лаура и Рики. Лаура харесва Джон. Питър харесва Лаура, но Лаура него – не . Рики работи зад гърба на Питър и може да види екрана му. Питър не заключва монитора си, когато ходи до кухнята в офиса. Вечерта преди да изтече инфромацията очевидци са видели Лаура да говори с братовчед си Сам (Sam). Сам работи за конкурентната фирма „Местен Сапунен Софтуер“. Софи(Sophie) и Сам са колеги. Лаура и Рики постоянно си говорят в офиса в кухнята.

Засечен е email изпратен от Сара до Рики. Има свидетели, които са видели Рики и Софи да обядват заедно. Рики харесва Софи, но тя не откликва на чувствата му.

На Рисунка 3.1 е онагледена случката. Изобразени са само базовите факти, а допълнителните ще бъдат изложени по-надолу поетапно.



Рисунка 3.1

4. Онтология на затворен свят

За да моделираме случката можем да използваме OWL DL. Ще представим таблично класовете и атрибутите към тях, описващи само това, което ние е необходимо за да намерим теча на информация. В началото ние имаме базовите класове, и към тях добавяме още два класа LeakInfoSourcePeople и LeakInfoDestinationPeople, които са рестрикции по дадени правила. Именно чрез тях ще можем лесно да формулираме някои заявки.

4.1 Елементи на онтологията

В таблица 4.1.1 са дадени класовете на началната онтология, според описаната история в т.3.

OWL клас	Родителски клас и описание
Person	Thing
Man	Person
Woman	Person, complementOf (Man)
Company	Thing
Project	Thing
Event	Thing

WitnessedEvent	Event
Conversation	WitnessedEvent
OnlineConversation	Conversation
Email	OnlineConversation
LeakInfoSourcePeople	owl:equivalentClass [rdfs:subClassOf sc:Person ; rdf:type owl:Restriction; owl:hasValue sc:roboSoapStrategyProject; owl:onProperty sc:worksFor]
LeakInfoDestinationPeople	owl:equivalentClass [rdf:type owl:Restriction; rdfs:subClassOf sc:Person ; owl:hasValue sc:domesticSoapSoftwareCompany; owl:onProperty sc:worksFor]

Таблица 4.1.1

В таблица 4.1.2 са описани свойствата на онтологията.

Property	Domain	Range	Characteristics
worksFor	Person	Company, Project	
hasConversationParticipant	Conversation	Person	*inverseOf* isPartOfConversation
isPartOfConversation	Person	Conversation	*inverseOf* hasConversationParticipant
canSeeMonitor	Person	Person	-
isGoodFriendTo	Person	Person	*symmetric*
canShareSensitiveInfo	Person	Person	TransitiveProperty
loveButNotLoved	Person	Person	*functional*

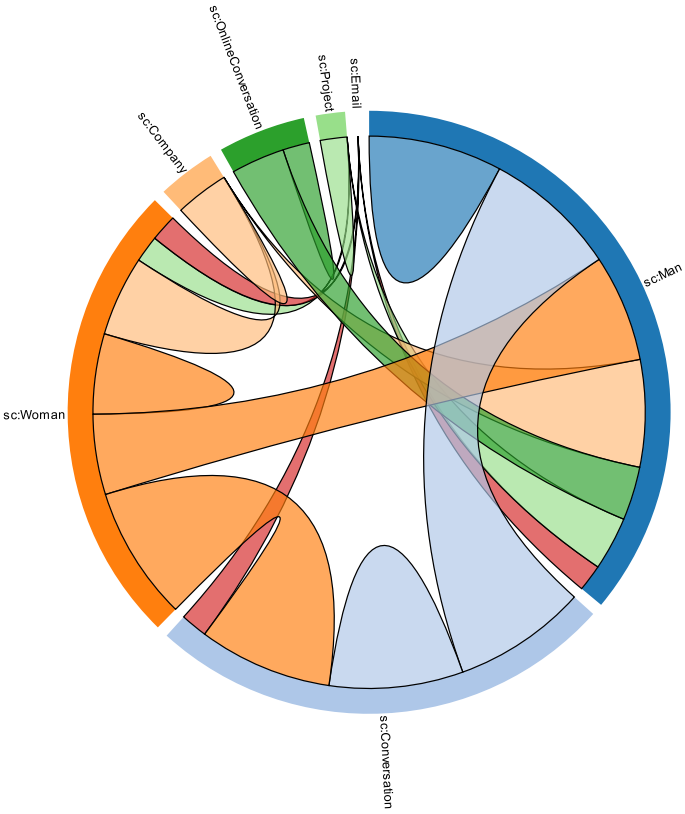
Таблица 4.1.2

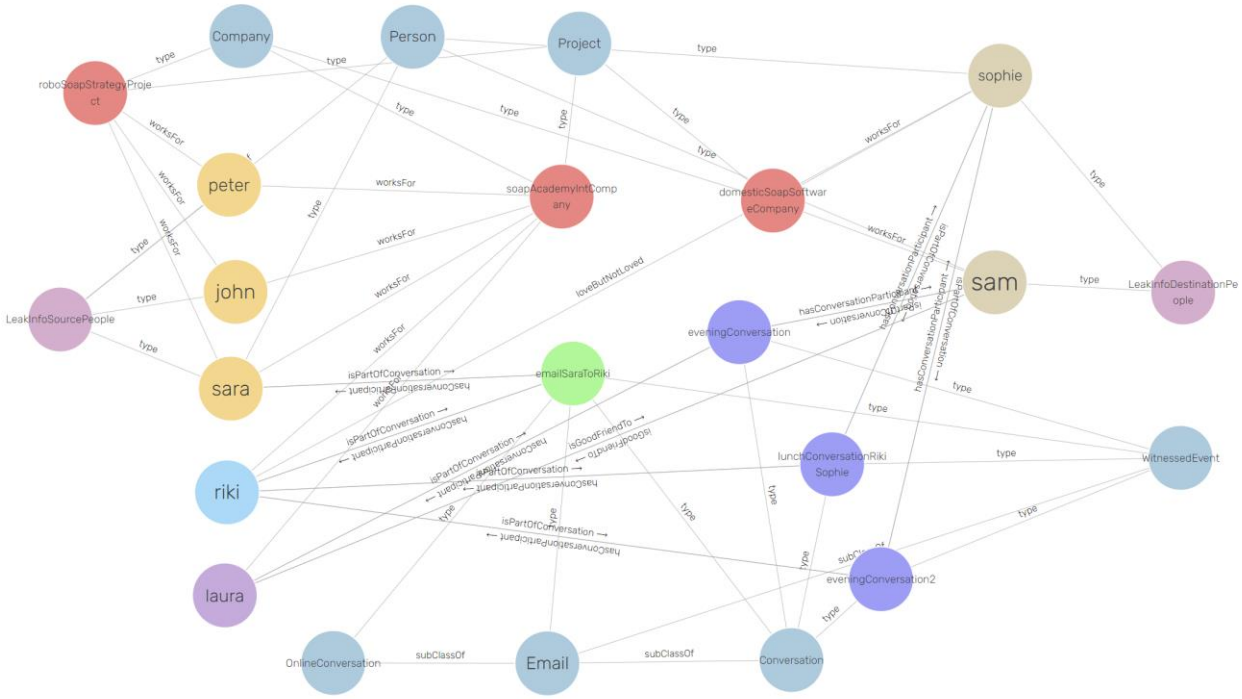
В Таблица 4.1.3 са дадени индивидите използвани в случката и техните свойства. Ограничили сме описанието само до ниво, за което можем да направим съответния извод. Например това, кои хора работят в една сграда, не е необходимо да се описва.

sc:john rdf:type sc:Man ; sc:worksFor sc:roboSoapStrategyProject,sc:soapAcademyIntCompany.
sc:sara rdf:type sc:Woman ; sc:worksFor sc:roboSoapStrategyProject, sc:soapAcademyIntCompany.
sc:peter rdf:type sc:Man; sc:loveButNotLoved sc:laura ; sc:worksFor sc:roboSoapStrategyProject,sc:soapAcademyIntCompany.
sc:riki rdf:type sc:Man ; sc:canSeeMonitor sc:peter ; sc:worksFor sc:soapAcademyIntCompany .
sc:sam rdf:type sc:Man ; sc:worksFor sc:domesticSoapSoftwareCompany .
sc:sophie rdf:type sc:Woman ; sc:worksFor sc:domesticSoapSoftwareCompany .
sc:soapAcademyIntCompany rdf:type sc:Company .
sc:domesticSoapSoftwareCompany rdf:type sc:Company .
sc:roboSoapStrategyProject rdf:type sc:Project .
sc:eveningConversation rdf:type sc:Conversation ; sc:hasConversationParticipant sc:sam , sc:laura .
sc:kitchenConversation rdf:type sc:Conversation ; sc:hasConversationParticipant sc:riki , sc:laura .
sc:peterAndJohnChat rdf:type sc:OnlineConversation ; sc:hasConversationParticipant sc:john , sc:peter .
sc:emailSaraToRiki rdf:type sc:Email ; sc:hasConversationParticipant sc:sara , sc:riki .
sc:lunchConversationRikiSophie rdf:type sc:Conversation ; sc:hasConversationParticipant sc:riki , sc:sophie .

Таблица 4.1.3

Ще представим и визуализация на онтологията на фигура 4.1.4.


sc:OnlineConversation
sc:Project
sc:Email
sc:Company
sc:Man
sc:Woman
sc:Conversation


Company
Person
Project
sophie
roboSoapStrategyProje ct
peter
soapAcademyIntComp any
domesticSoapSoftwar eCompany
sam
LeakinfoDestinationPe ople
john
LeakinfoSourcePeople
eveningConversation
loveButNotLoved
sara
emailSaraToRiki
riki
lunchConversationRiki Sophie
witnessedEvent
laura
eveningConversation2
OnlineConversation
Email
Conversation
type
worksFor
subClassOf

Фигура 4.1.4

За да се направи качествен извод, от къде би могло да изтече информация, са необходими определени заявки на SPARQL.

4.2. Изводи със SPARQL[4] към базата

Като извод искаме да намерим всички които биха могли да споделят информация в разговор или събитие. Можем да извлечем всички подкласове на „sc:Event“.

```
# sparql/sparql1.txt
PREFIX sc: <urn:soap/crime/ontology#>
PREFIX owl:  <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

select *
where {
    ?c a owl:Class .
    ?c rdfs:subClassOf sc:Event .
}
```

	c
1	owl:Nothing
2	sc:Conversation
3	sc:OnlineConversation
4	sc:Event
5	sc:WitnessedEvent
6	sc:Email

Таблица 4.2.1

Нормално е да виждаме „owl:Nothing“ тъй като той е подклас на всеки клас. За останалите трябва да ги видим като граф за да добием представа. За да видим как е структурирана йерархията ще направим друга заявка, в която ще искаме да има и родителския клас и ще филтрираме за ненужните класове като owl:Thing.

```
# sparql/sparql2.txt
PREFIX sc: <urn:soap/crime/ontology#>
PREFIX owl:  <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

select *
where {
    ?c a owl:Class .
    ?c rdfs:subClassOf sc:Event .
    ?c rdfs:subClassOf ?parent .
    FILTER (?c != owl:Nothing)
    FILTER (?parent != owl:Thing)
    FILTER (!sameTerm(?c, ?parent))
}
```

	C	parent
1	sc:Conversation	sc:Event
2	sc:Conversation	sc:WitnessedEvent

3	sc:OnlineConversation	sc:Conversation
4	sc:OnlineConversation	sc:Event
5	sc:OnlineConversation	sc:WitnessedEvent
6	sc:WitnessedEvent	sc:Event
7	sc:Email	sc:Conversation
8	sc:Email	sc:OnlineConversation
9	sc:Email	sc:Event
10	sc:Email	sc:WitnessedEvent

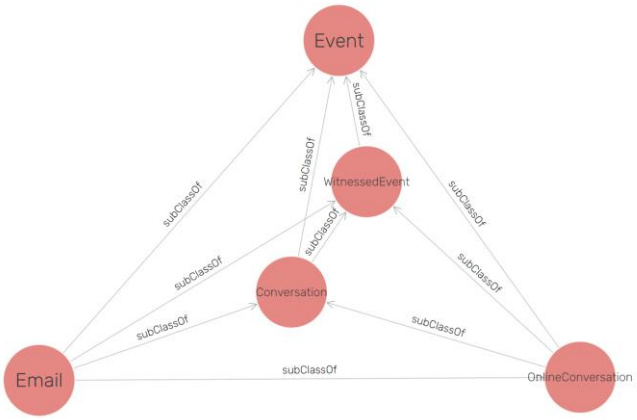
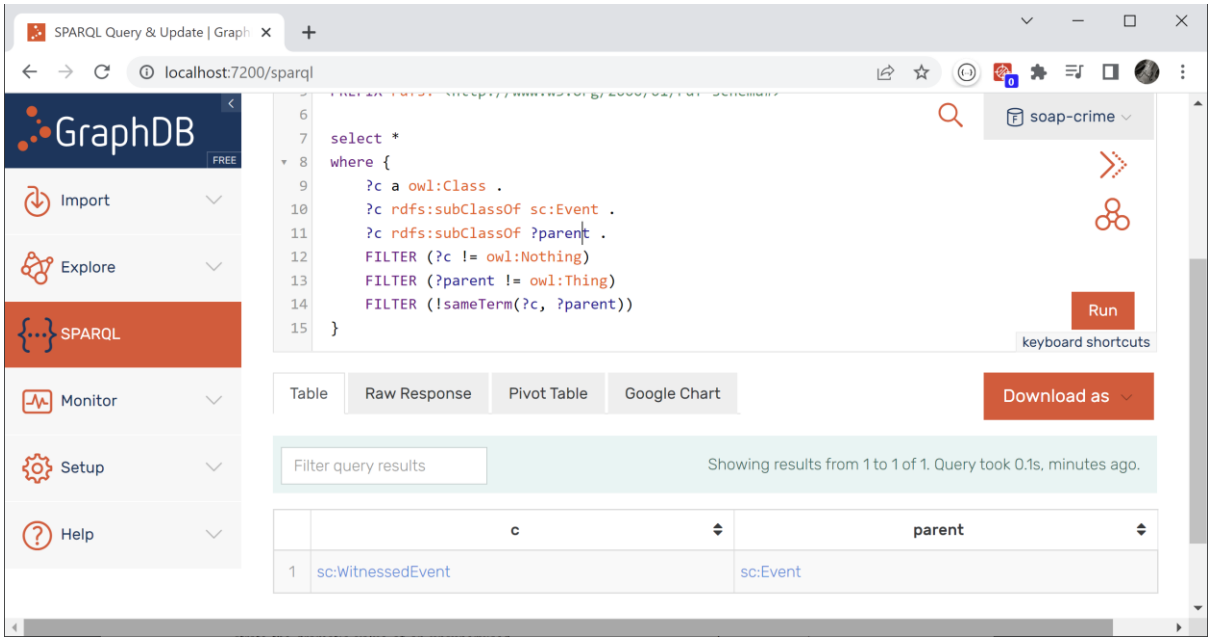


Таблица 4.2.2

Тази йерархия е построена на базата на OWL машината за изводи. Ако изключим машината за изводи бихме получили само директния наследник на „sc:Event“. Даваме пример в следващата фигура при изключени изводи.



Фигура 4.2.1

За целта знаем, че всички хора, които имат секретната информация са от клас sc:LeakInfoSourcePeople. Нека да видим следната заявка.

```
# sparql/sparql3.txt
PREFIX sc: <urn:soap/crime/ontology#>
select * where {
    ?p1 a sc:LeakInfoSourcePeople .
    ?p1 sc:isPartOfConversation ?c.
    ?p2 sc:isPartOfConversation ?c.
    FILTER (!sameTerm(?p1, ?p2))
}
```

	p1	c	p2
1	sc:peter	sc:peterAndJohnChat	sc:john
2	sc:john	sc:peterAndJohnChat	sc:peter
3	sc:sara	sc:emailSaraToRiki	sc:riki

Таблица 4.2.3

В случая не ни води надалеч. Нека да изведем и графично какво представлява резултата, използвайки бутона VISUAL от уеб-клиента за GraphDB.

```
# sparql/sparql4.txt
PREFIX sc: <urn:soap/crime/ontology#>
construct {
    ?p1 a sc:Person .
    ?p2 a sc:Person.
} where {
  ?p1 a sc:LeakInfoSourcePeople .
  ?p1 sc:isPartOfConversation ?c.
  ?p2 sc:isPartOfConversation ?c.
  FILTER (!sameTerm(?p1, ?p2))
}
```

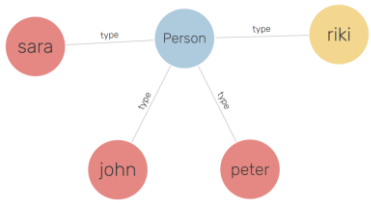


Таблица 4.2.4

Нека да разширим заявката и проследим и всички които са участвали в разговори тръгващи от хора с клас sc:LeakInfoSourcePeople.

```
# sparql/sparql5.txt
PREFIX sc: <urn:soap/crime/ontology#>

select * where {
    ?p1 a sc:LeakInfoSourcePeople .
    ?p1 (sc:isPartOfConversation/sc:hasConversationParticipant)* ?p2
    FILTER (!sameTerm(?p1, ?p2))
}
```

```
        p1                  p2
1       sc:peter        sc:john
2       sc:john         sc:peter
3       sc:sara         sc:riki
4       sc:sara         sc:laura
5       sc:sara         sc:sophie
6       sc:sara         sc:sam
```

Таблица 4.2.5

Сега можем да сложим и ограничение да ни води само до хора от класа „sc:LeakInfoDestinationPeople“ за да намалим резултатите.

```
# sparql/sparql6.txt
PREFIX sc: <urn:soap/crime/ontology#>

select * where {
    ?p1 a sc:LeakInfoSourcePeople .
    ?p2 a sc:LeakInfoDestinationPeople.
    ?p1 (sc:isPartOfConversation/sc:hasConversationParticipant)* ?p2
}
```

```
        p1      p2
1       sc:sara sc:sam
2       sc:sara sc:sophie
```

Таблица 4.2.6

Тук можем да отбележим, че не включваме всички възможности да се пренесе информация. Какво сме пропуснали? Детективът има отбелязани само разговори, които са със свидетели или записани, но може да има и такива, за които никой не знае. Това са разговорите между добрите приятели. Така можем да включим още едно разклонение за проследяване на информацията между ?p1 и ?p2.

```
# sparql/sparql7.txt

PREFIX sc: <urn:soap/crime/ontology#>
select * where {
    ?p1 a sc:LeakInfoSourcePeople .
    ?p2 a sc:LeakInfoDestinationPeople.
    ?p1
((sc:isPartOfConversation/sc:hasConversationParticipant)|sc:isGoodFriendT
o)* ?p2
}
```

```
        p1      p2
1       sc:sara sc:sam
2       sc:sara sc:sophie
```

Таблица 4.2.7

Резултатът не се промени. Има и още един начин да се предаде информация. Колегите могат да видят монитора на тези, които не заключват екраните си. Можем да включим и това като допълнително условие, но със символа „^“ защото търсим обратно насочена дъга за пренос на инфорация, а именно към ?p2.

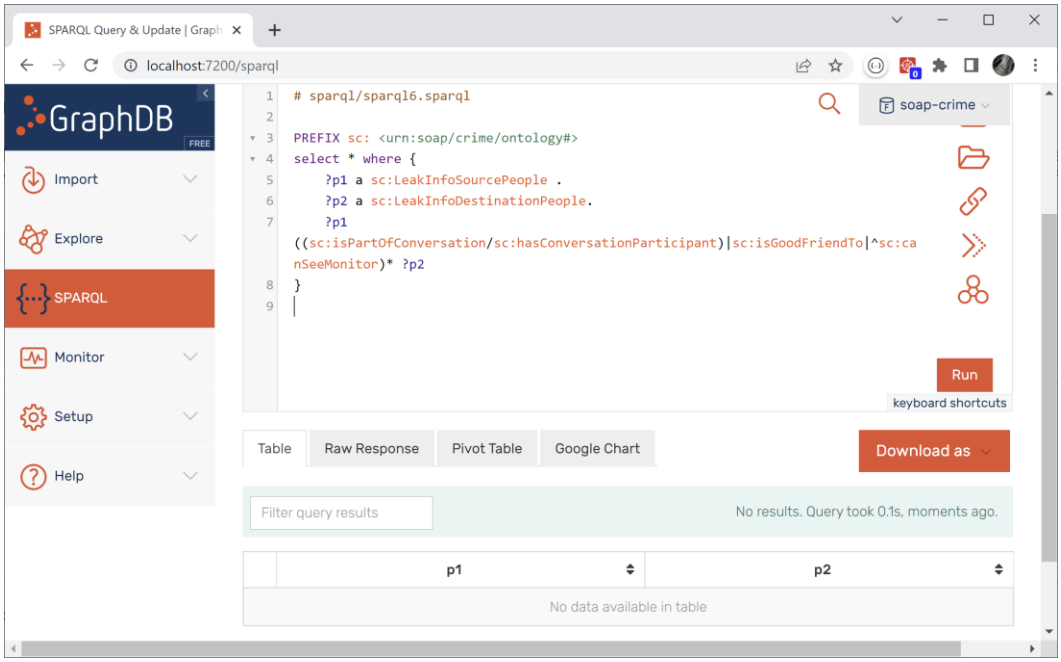
```
# sparql/sparql8.txt

PREFIX sc: <urn:soap/crime/ontology#>
select * where {
    ?p1 a sc:LeakInfoSourcePeople .
    ?p2 a sc:LeakInfoDestinationPeople.
    ?p1
((sc:isPartOfConversation/sc:hasConversationParticipant)|sc:isGoodFriendT
o|^sc:canSeeMonitor)* ?p2
}
```

	p1	p2
1	sc:sara	sc:sam
2	sc:peter	sc:sam
3	sc:john	sc:sam
4	sc:sara	sc:sophie
5	sc:peter	sc:sophie
6	sc:john	sc:sophie

Таблица 4.2.8

Излязоха твърде много резултати, това са твърде много хипотези, които трябва да бъдат проверени. Нека да изключим мащината за изводи на базата и да видим какво ще се получи като резултат.



Фигура 4.2.2

Разбира се това е празен резултат. Тоест това ни дава увереност, че машината за изводи ни помага реално да разберем какво се е случило.

Сега искаме да разберем през какви разговори е излязла информацията на от първа ръка, това вероятно ще ни помогне да елиминираме част от хипотезите.

```
# sparql/sparql9.txt

PREFIX sc: <urn:soap/crime/ontology#>
select ?p1 ?firstHop ?p2
where {
    ?p1 a sc:LeakInfoSourcePeople .
    ?p2 a sc:LeakInfoDestinationPeople.
    ?p1 sc:isPartOfConversation ?firstHop .
    ?firstHop
((sc:hasConversationParticipant)|sc:isGoodFriendTo|^sc:canSeeMonitor)*
?p2 .
}
```

	p1	firstHop	p2
1	sc:peter	sc:peterAndJohnChat	sc:sam
2	sc:john	sc:peterAndJohnChat	sc:sam
3	sc:sara	**sc:emailSaraToRiki**	sc:sam

Таблица 4.2.9

Тук можем да видим, че без да имаме предвид изтичане на информация чрез приятелството и гледането на монитор, имаме един имейл. Детектива се заема да попита администратора на имейл сървъра и този имейл може да бъде проверен дали има чувствителна информация. Няма проблем с имейла и не е имало теч от него, така че можем да го елиминираме като изходящи имейли от първа ръка.

```
# sparql/sparql10.txt
PREFIX sc: <urn:soap/crime/ontology#>

select *
where {
    ?p1 a sc:LeakInfoSourcePeople .
    ?p2 a sc:LeakInfoDestinationPeople.

    {
      ?p1 sc:isPartOfConversation ?firstHopConversation .
      ?firstHopConversation
((sc:hasConversationParticipant)|sc:isGoodFriendTo|^sc:canSeeMonitor)+
?p2 .
      FILTER (!sameTerm(?p1, ?p2))
      FILTER NOT EXISTS{?firstHopConversation rdf:type sc:Email. }
    } UNION  {
      ?p1 (sc:isGoodFriendTo|^sc:canSeeMonitor) ?firstHopPerson .
```

```
      ?firstHopPerson
((sc:isPartOfConversation/sc:hasConversationParticipant)|sc:isGoodFriendT
o|^sc:canSeeMonitor)+ ?p2 .
      FILTER (!sameTerm(?p1, ?p2))
    }
}
```

	p1	p2	firstHopConversation	firstHopPerson
1	sc:peter	sc:sam	sc:peterAndJohnChat	
2	sc:john	sc:sam	sc:peterAndJohnChat	
3	sc:peter	sc:sophie		sc:riki
4	sc:peter	sc:sam		sc:riki

Таблица 4.2.10

В горната заявка правим обхождане по два пътя към целта. В първият вариант искаме всички разговори от които първият не е имейл. Във вторият искаме обхождане тръгвайки от въпросният човек но на първа инстанция нямаме разговор а имаме условието „sc:isGoodFriendTo|^sc:canSeeMonitor“, което означава да се последват дъги от графа които са приятел или човека от резултатния възел да вижда монитора на източника. Двата резултата се обединяват с UNION. Виждаме, че имаме само 4 хипотези спрямо предните 6, тоест сме успели да ги редуцираме. Нека като следваща заявка да обединим в удобен за разглеждане вариант хипотезите.

```
# sparql/sparql11.txt
PREFIX sc: <urn:soap/crime/ontology#>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
select ?hypothesis
where {
    ?p1 a sc:LeakInfoSourcePeople .
    ?p2 a sc:LeakInfoDestinationPeople.
    { ?p1 sc:isPartOfConversation ?firstHopConversation .
      ?firstHopConversation
((sc:hasConversationParticipant)|sc:isGoodFriendTo|^sc:canSeeMonitor)+
?p2 .
      FILTER (!sameTerm(?p1, ?p2))
      FILTER NOT EXISTS{?firstHopConversation rdf:type sc:Email. }
    } UNION  {
      ?p1 (sc:isGoodFriendTo|^sc:canSeeMonitor) ?firstHopPerson .
      ?firstHopPerson
((sc:isPartOfConversation/sc:hasConversationParticipant)|sc:isGoodFriendT
o|^sc:canSeeMonitor)+ ?p2 .
      FILTER (!sameTerm(?p1, ?p2))
    }
    BIND (
        IF(bound(?firstHopConversation),
           STRAFTER(str(?firstHopConversation),'#'),
           STRAFTER(str(?firstHopPerson),'#'))
        as ?firstHop)
    BIND (concat(STRAFTER(str(?p1),'#'),
            '-',
            ?firstHop,
```

```
            '-*-',
            STRAFTER(str(?p2),'#'))
        as ?hypothesis)
}
```

```
        hypothesis
1 "peter-peterAndJohnChat-*-sam"
2 "john-peterAndJohnChat-*-sam"
3 "peter-riki-*-sam"
4 "peter-riki-*-sophie"
```

Таблица 4.2.11

Сега хипотезите за проверка от детектива са ясни и са само 4 на брой. Бихме могли да използваме и знанието за loveButNotLoved но това би ни подвело, защото няма как да проверим междинните възли. Нека все пак да видим пример.

```
# sparql/sparql12.txt
PREFIX sc: <urn:soap/crime/ontology#>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
select ?hypothesis ?x
where {
        ?p1 a sc:LeakInfoSourcePeople .
    ?p2 a sc:LeakInfoDestinationPeople.
    ?x sc:loveButNotLoved ?p2 .
    { ?p1 sc:isPartOfConversation ?firstHopConversation .
      ?firstHopConversation
((sc:hasConversationParticipant)|sc:isGoodFriendTo|^sc:canSeeMonitor)+
?p2 .
      FILTER (!sameTerm(?p1, ?p2))
      FILTER NOT EXISTS{?firstHopConversation rdf:type sc:Email. }
    } UNION  {
      ?p1 (sc:isGoodFriendTo|^sc:canSeeMonitor) ?firstHopPerson .
      ?firstHopPerson
((sc:isPartOfConversation/sc:hasConversationParticipant)|sc:isGoodFriendT
o|^sc:canSeeMonitor)+ ?p2 .
      FILTER (!sameTerm(?p1, ?p2))
    }
    BIND (
        IF(bound(?firstHopConversation),
           STRAFTER(str(?firstHopConversation),'#'),
           STRAFTER(str(?firstHopPerson),'#'))
        as ?firstHop)
    BIND (concat(STRAFTER(str(?p1),'#'),
            '-',
            ?firstHop,
            '-*-',
            STRAFTER(str(?p2),'#'))
        as ?hypothesis)
}
```

```
        hypothesis     x
1 "peter-riki-*-sophie"  sc:riki
```

Таблица 4.2.12

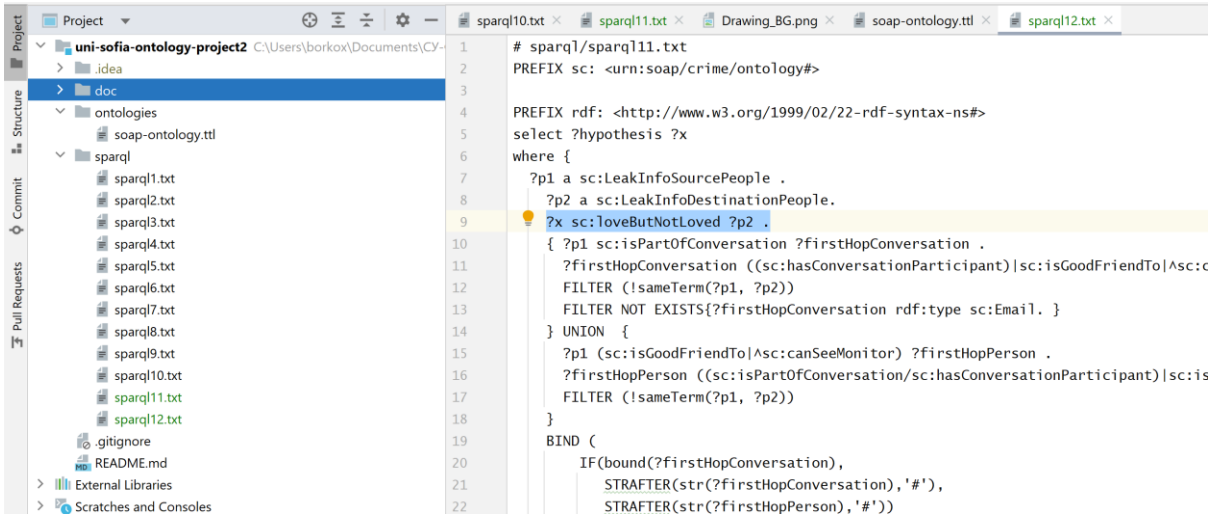
Тук разбира се излиза една хипотеза, но все още са възможни и останалите 3 от Таблица 4.2.11. Случката е формулирана без еднозначен отговор, както обикновено се случва в реалния живот.

4.3. Анализ на резултатите

До момента използвахме SPARQL за да видим началото на изтичане на информация и края на изтичане на информация. Междинните звена обаче също са важни и видяхме как може да бъде извлечено първата крачка от една верига в графа. Извеждане на пътя от ?p1 до ?p2 не е лека задача и може да се използват различни техники. Може да се използва „sc:canShareSensitiveInfoTo“ и да се наслага като атрибут между участниците и тогава вероятно малко ще се улесни задачата. Друг подход е да се използват правила за GraphDB като файл „*.pie“, но това е свързано с доста специфични знания за базата и е нелека задача. За целите на проекта, отговор на въпроса има ли теч и възможните хипотези, имаме Таблица 4.2.11 и тя представлява отправната точка за вземане на решение в реална ситуация.

5. Реализация на проекта

Общата структура на проекта са папки с текстово описание на онтологията като TTL (Turtle format), файлове описващи заявките на SPARQL. Сървър на който се импортира онтологията е GraphDB. Файловете се разглеждат и редактират с IntelliJ, но всеки друг текстов редактор би свършил работа. Линк към сорс кода е качен в гитхъб (Вж. Приложение 1) и е неразделна част от този документ. Структурата на приложението е дадена на фигура 5.1. Използваната среда за текстообработка и работа с git е IntelliJ .



Фигура 5.1

Подробни инструкции са дадени в README.md файла.

6. Идеи за бъдещо развитие и подобрения

До момента не използвахме релацията „sc:canShareSensitiveInfoTo“. Първоначалната ми идея беше да насложа тази релация с редица SPARQL INSERT заявки и така постепенно да анотирам графа с хората, които биха могли да споделят секретната информация, но тогава нямаше да мога да демонстрирам силата на SPARQL. Сега си мисля, че би било добра идея и това да се направи, което ще разшири доста обема на текущата работа, но е едно възможно подобрение.

В заключение ще кажа, че изложеният пример на криминална случка е само една демонстрация на технологията в един неин вариант. Би могло проекта да се адаптира към по-общ модел на данни, вкарвани или извличани по автоматизиран начин.

7. Източници и използвана литература

[1] Knowledge Representation and Reasoning, Ronald Brachman, Hector Levesque, 2004, The Morgan Kaufmann Series in Artificial Intelligence

[2] GrapgDB documentation, Ontotext 2015-2022, https://graphdb.ontotext.com/documentation/enterprise/

[3] Web Ontology Language (OWL), W3C, https://www.w3.org/OWL/

[4] SPARQL By Example, W3C, https://www.w3.org/2009/Talks/0615-qbe/

Приложения

1. Сорс код (Source code)

https://github.com/borkox/uni-sofia-ontology-project2